# BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images, please do not report the images to the Image Problems Mailbox.**

**PCT** WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04Q — A2

(11) International Publication Number: WO 97/48234

(43) International Publication Date: 18 December 1997 (18.12.97)

(21) International Application Number: PCT/US97/09996

(22) International Filing Date: 9 June 1997 (09.06.97)

(30) Priority Data:
08/662,077 12 June 1996 (12.06.96) US

(71) Applicant: NEWNET, INC. [US/US]; 2 Enterprise Drive, Shelton, CT 06484 (US).

(72) Inventors: HARTMANN, Curtis; 308 Shore Drive, Branford, CT 06405 (US). DUMAN, Osman, Ali; 30 Wake Robin Lane, Shelton, CT 06484 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

**Published**
*Without international search report and to be republished upon receipt of that report.*

(54) Title: METHODS AND APPARATUS FOR CONTROLLING DIGITAL COMMUNICATIONS SWITCHING EQUIPMENT

(57) Abstract

A "generic" switch messaging protocol is disclosed for message handling and switch supervision in conjunction with a number of switching engines, each of which is conversant with the generic messaging protocol, each switching engine also being conversant with a specific switch messaging protocol. An object oriented development system is also disclosed utilizing a "generic" switch messaging protocol and a plurality of switching engines, each of which is conversant with the generic messaging protocol and each of which is conversant with a specific switch messaging protocol. Certain switch messages are not "genericized" because their functionality is different from the functionality of other switches. These messages generally include initialization and maintenance messages which are hardware specific and have no counterpart in another switch from a different vendor. In order to handle these messages, specific data files are provided in the switching engine for automatic download to the switch as well as a specific MML for interpreting configuration commands. Also, according to the invention, some commands or messages which are not otherwise supported by a particular switch can nevertheless be supported in the API by providing the switch engine with the intelligence to combine native switch messages to "emulate" a functionality which is not directly provided by the switch.

*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| AL | Albania | ES | Spain | LS | Lesotho | SI | Slovenia |
| AM | Armenia | FI | Finland | LT | Lithuania | SK | Slovakia |
| AT | Austria | FR | France | LU | Luxembourg | SN | Senegal |
| AU | Australia | GA | Gabon | LV | Latvia | SZ | Swaziland |
| AZ | Azerbaijan | GB | United Kingdom | MC | Monaco | TD | Chad |
| BA | Bosnia and Herzegovina | GE | Georgia | MD | Republic of Moldova | TG | Togo |
| BB | Barbados | GH | Ghana | MG | Madagascar | TJ | Tajikistan |
| BE | Belgium | GN | Guinea | MK | The former Yugoslav Republic of Macedonia | TM | Turkmenistan |
| BF | Burkina Faso | GR | Greece | | | TR | Turkey |
| BG | Bulgaria | HU | Hungary | ML | Mali | TT | Trinidad and Tobago |
| BJ | Benin | IE | Ireland | MN | Mongolia | UA | Ukraine |
| BR | Brazil | IL | Israel | MR | Mauritania | UG | Uganda |
| BY | Belarus | IS | Iceland | MW | Malawi | US | United States of America |
| CA | Canada | IT | Italy | MX | Mexico | UZ | Uzbekistan |
| CF | Central African Republic | JP | Japan | NE | Niger | VN | Viet Nam |
| CG | Congo | KE | Kenya | NL | Netherlands | YU | Yugoslavia |
| CH | Switzerland | KG | Kyrgyzstan | NO | Norway | ZW | Zimbabwe |
| CI | Côte d'Ivoire | KP | Democratic People's Republic of Korea | NZ | New Zealand | | |
| CM | Cameroon | | | PL | Poland | | |
| CN | China | KR | Republic of Korea | PT | Portugal | | |
| CU | Cuba | KZ | Kazakstan | RO | Romania | | |
| CZ | Czech Republic | LC | Saint Lucia | RU | Russian Federation | | |
| DE | Germany | LI | Liechtenstein | SD | Sudan | | |
| DK | Denmark | LK | Sri Lanka | SE | Sweden | | |
| EE | Estonia | LR | Liberia | SG | Singapore | | |

# METHODS AND APPARATUS FOR CONTROLLING DIGITAL COMMUNICATIONS SWITCHING EQUIPMENT

This application is a continuation-in-part of application Serial Number 08/555,040 filed 11/8/95 entitled System Permitting User Defined Managed Objects for Configuration of Telecommunications Equipment, the complete disclosure of which is incorporated by reference herein.

## BACKGROUND OF THE INVENTION

### 1. Field of the Invention

The invention relates to high capacity digital telecommunications switches which are controlled by a host. More particularly, the invention relates to compatible methods for controlling otherwise incompatible digital telecommunications switches as well as apparatus incorporating the methods.

### 2. State of the Art

Modern telecommunications systems utilize high capacity digital switching systems to process calls and effect different types of telecommunications. These switches are typically built in a modular form utilizing a bus and a plurality of modular cards which are mounted in racks and slots. A typical switch is the LNX-2000 from Excel, Inc., Sagamore Beach, MA. Prior art Figure 1 shows the general physical configuration of the LNX-2000 and prior art Figure 2 shows a general block circuit diagram illustrating a bus and a plurality of modular cards.

Referring now to Figures 1 and 2, the digital switch 10 includes a rack 12 having slots 14a-14t which are associated with a bus 16. Typically, a redundant bus 16a is also provided as a back-up should the bus 16 fail for any reason. The first two slots 14a, 14b are usually reserved for a power supply 18 and a redundant power supply 18a. The last two slots 14s, 14t are usually reserved for a switch matrix CPU 20 and a redundant switch matrix CPU 20a. Each matrix CPU is provided with a respective control link 22, 22a for connection to a host 40. The

control links are usually RS-232 serial links. Each matrix CPU may also be provided with a reference clock port (not shown). The other slots are provided for modular cards 24, 26, 28, 30, and 32a-k, for example. The cards may provide access telecommunications channels, provide telecommunications signalling, or provide other telecommunications services. For example, card 24 shown in Figure 2 provides a 192-channel T1 interface. Card 26 provides a 256-channel E1 interface. Card 28 provides a 24-channel ISDN Primary Rate interface. Card 30 provides digital signal processing such as tone generation, tone reception, digital voice recording and playback, etc. It will be appreciated that some of the interface cards will rely on functions of other cards in order to execute certain telecommunications functions. In general, each of the cards will rely on the matrix CPU for intelligent control of their basic functions. In addition, the T1 interfaces and the E1 interfaces may cooperate to provide conversions. The ISDN interfaces will rely on the T1 or E1 interfaces for B channel access. The DSP cards may be called upon by all of the interface cards to generate appropriate signalling tones.

Each of the cards in switch 10 is typically configured to operate in a user definable manner. Configuration of the cards is effected by the host 40 which sends configuration data to the matrix CPU 20 (20a) which in turn remembers the configuration data for each card in the switch. In addition, the matrix CPU contains system software which is downloaded to it by the host. The system software governs the basic switch operation. Call processing activities are controlled by the host which remains in communication with the matrix CPU so long as the switch is in service.

Typical switch configuration commands issued by the host to the switch (the matrix CPU) include: resetting line cards, changing the service state of a card (placing it in and out of service), resetting the matrix CPU, downloading system software to the matrix CPU, setting the system time, system network synchronization, assigning logical span IDs, changing a channel

service state, changing a channel configuration, changing a trunk type configuration (defining the signalling protocol for a particular channel or group of channels), changing a start dialing signalling protocol, configuring timers and filters (for signalling), busying out trunks, setting answer supervision mode(for each channel), setting release modes for a channel, setting call setup inpulsing parameters. Once configured, the switch remains configured until a card or the matrix CPU is reset, or until a change in service dictates changing one or more of the configuration parameters. For example, if new trunks are added or new cards are added, additional configuration will be required. Usually, the configuration data is saved as a file by the host and downloaded to the matrix CPU together with system software if the switch is reset.

Typical ongoing communication between the switch (the matrix CPU) and the host includes call processing, control of tone generation and reception, and call progress analysis. Call processing relates to how the switch handles incoming and outgoing calls and includes the functions of call setup, cross connections, inseize/outseize call control, incoming call setup, inseize control instructions, outgoing call setup, outseize control instructions and connection management. Tone generation control includes defining tone specifications, tone detection, digit collection, tone generation, outpulsing setup, and generating tones for prompting and call status information (e.g. dial tones and busy signals). Call progress analysis is generally used when originating a call to the PSTN and includes the functions of invoking the analysis, designing the analysis, configuring global parameters and cadence pattern parameters, setting pattern defaults and pattern IDs and setting class and tone group defaults.

The LNX 2000 Switch provides a host messaging protocol and specific message formats for communication between a host and the switch. Other switches provide their own host messaging protocols which deal with similar activities but which have different message formats. Thus, a host must be programmed in

one way to control one brand of switch and in a different way to control another brand of switch. That is, different switches use different hardware components and therefore respond to different message sets. For example, the Summa 4 SDS Series of switches includes a Network Bus Controller Card, a Bus Repeater Card, a Direct Inward Dial Card, an E&M Trunk Card, a Subscriber Line Interface Card, a Universal Trunk Card, a Digital Conference Card, among others. These cards require different messages for configuration and operation than the coards mentioned above with regard to the LNX 2000 Switch. Moreover, the SDS Series utilizes its own Unix-based command language which is different from the command language used by the LNK 2000 Switch.

Parent application Serial Number 08/555,040 discloses an object oriented development system for the configuration of telecommunications equipment including one or more digital telecommunications switches. Figure 3 illustrates a graphical representation of the structure of the development system. The development system includes an object server 50 having at least one man/machine interface (MMI) agent 52, an object server 54 with predefined managed objects 56 and a database management library 58, a server applications programmer interface (API) 60 coupled to the MMI and the object server for hiding the internal architecture of the object services from the MMI agent with respect to the managed objects 56, and means for permitting the developer to create and define user-defined managed objects 57 which utilize the database management library and the database which stores managed object related data, and which does not require the server API to be rewritten. The server applications programmer interface (API) is organized and designed in a manner such that the user defined objects are automatically supported by the API. Predefined managed objects are initialized at start-up before the initialization of the user-defined objects. The development environment allows for the rapid development and deployment of new telecommunications services using combinations of predefined managed objects and user defined managed objects. The managed objects can include hardware such as shelf, rack, board, switch, signalling, etc., service software such as call

processing triggers and features, configuration software such as rule group and alarm, and user defined objects. As shown in Figure 3, a hardware object 56a is provided to interface with a high speed digital switch so that applications developed in the system 50 can act as a host to the switch. As mentioned above, however, switches from different manufacturers communicate using different protocols. This makes object definition more complicated because the switch signalling and supervision messages of the API of the system 50 must be rewritten to accommodate the protocol of different switches.

## SUMMARY OF THE INVENTION

It is therefore an object of the invention to provide a method by which a single API can be used to control a number of switches having different message protocols.

It is also an object of the invention to provide modular switching engines for use with an object oriented development system for the configuration of telecommunications equipment including one or more digital telecommunications switches.

It is another object of the invention to provide an object oriented development system for the configuration of telecommunications equipment including one or more digital telecommunications switches which includes a number of switching engines so that a number of switches utilizing different messaging protocols can be controlled from a single API.

It is still another object of the invention to provide methods and apparatus whereby a single host computer can be used to control a plurality of different switches, each switch having a different messaging protocol.

In accord with these objects which will be discussed in detail below, the methods of the present invention include providing a "generic" switch messaging protocol for message handling and switch supervision and providing a number of

switching engines, each of which is conversant with the generic messaging protocol, each switching engine also being conversant with a specific switch messaging protocol. The apparatus according to the invention includes an object oriented development system utilizing a "generic" switch messaging protocol and a plurality of switching engines, each of which is conversant with the generic messaging protocol and each of which is conversant with a specific switch messaging protocol. According to the invention, certain switch messages are not "genericized" because their functionality is different from the functionality of other switches. These messages generally include initialization and maintenance messages which are hardware specific and have no counterpart in another switch from a different vendor. In order to handle these messages, specific data files are provided in the switching engine for automatic download to the switch as well as a specific MML for interpreting configuration commands. Also, according to the invention, some commands or messages which are not otherwise supported by a particular switch can nevertheless be supported in the API by providing the switch engine with the intelligence to combine native switch messages to "emulate" a functionality which is not directly provided by the switch.

Additional objects and advantages of the invention will become apparent to those skilled in the art upon reference to the detailed description taken in conjunction with the provided figures.

## BRIEF DESCRIPTION OF THE DRAWINGS AND APPENDICES

Figure 1 is a perspective view of a prior art high speed digital switch;

Figure 2 is a block diagram of a prior art switch bus;

Figure 3 is a block diagram of an object oriented development system for the configuration of telecommunications equipment;

Figure 4 is a block diagram of a switching engine according to the invention; and

Figure 5 is a block diagram of an object oriented development system for the configuration of telecommunications equipment including a plurality of switching engines according to the invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In accord with the present invention, the functionality of a high speed digital switch is divided into four areas: media, in-band signalling, connection, and switch managed objects. Media refers to the ability to generate tones, collect digits, outpulse digits, and play recorded announcements. In-band signalling refers to the ability to perform conventional in-band signalling such as call progress analysis. Connection refers the ability to manage connections such as call processing. Switch managed objects refers to configuring and initializing boards, spans, ports, etc. A generic messaging protocol according to the invention is organized according to the first three of these four areas of functionality. The fourth area of functionality is serviced by a superset of commands which includes commands for configuring and maintaining a variety of switches. A switching engine according to the invention is provided with a media server, an in-band signalling server, a connection server, and a switch managed objects server.

Before explaining the specifics of the generic messaging protocol, it is useful to examine first the internal functionality of a switching engine according to the invention. Figure 4 shows an internal functional diagram of a switching engine 100 according to the invention. It will be appreciated from Figure 4 that the switching engine 100 communicates on the one hand with a particular brand of switch 102 and on the other hand with the Call Control entity (call processing application) 104 and the Man-Machine Interface 106 of a development system according to the invention which is described in detail below

with reference to Figure 5. The switching engine 100 is also provided with the ability to read from and write to persistent configuration files 108 which are particular to a given switch 102 and which contain configuration and maintenance functions.

Referring now to Figure 4 in detail, the switching engine 100 is provided with several translators, managers and interfaces. The first three areas of functionality mentioned above are handled by the Call Control entity 104. Therefore, the switching engine 100 is provided with a Connection API translator 110, a Media API translator 112, and an In-band Signalling API translator 114 for communicating with the Call Control entity 104. These translators provide the logic necessary to convert generic messages from the Call Control entity 104 into native messages for the particular switch 102 and to convert native messages from the switch to generic messages which are used by the Call Control 104. Accordingly, these translators communicate with the switch 102 via a Call Control Transaction Manager 116 and a Switch Message Interface 118. The Control Transaction Manager 116 provides the logic necessary to manage API transactions on a per-device basis. In this regard, a Logical Device Management 120 communicates with the Control Transaction Manager 116 and each of the translators mentioned above. The Logical Device Management 120 provides the logic necessary to manage per-device information. Finally, the Switch Message Interface 118 provides the logic necessary to communicate with the switching matrix. In this regard, an Alarm Interface 122 communicates with the Switch Message Interface 118 to convert native switch alarms to alarms which can be used by the development system of the invention.

The fourth area of functionality mentioned above is handled by the MMI 106 via a superset of commands which includes configuration and maintenance commands for several switches. Each switching engine 100 is provided with an Object Server Interface Translator 124 which provides the logic necessary to convert object server API messages from the MMI into native switch "operation, administration, and maintenance" (OA&M)

messages and to convert native OA&M messages into object server API messages. In order to manage OA&M messages on a per-device basis, the Object Server Interface Translator 124 also communicates with the Logical Device Management 120. Messages to and from the Object Server Interface Translator 124 pass through the Native Switch OA&M Transaction Manager 126 and the Switch OA&M 128 which provide the logic necessary to manage transactions with the switch 102 over an OA&M interface. As mentioned above, some of the OA&M messages for a given switch are stored in data files 108 and are used at the time the switch is initialized. These files are accessed by the Object Server Interface Translator 124 and an Initialization Manager 130, the latter of which provides the logic necessary to initialize the switch on cold start-up.

As mentioned above, the switching engines according to the invention all communicate with a generic message protocol of the invention and each switching engine communicates with a particular brand of digital switch. The diagram in Figure 4 is a general schematic diagram which applies to each of the several switching engines of the invention. The differences between the switching engines lie in how they deal with the generic messages from the development system of the invention. Therefore, in order to further elaborate on the functionality of each of the switching engines, it is necessary to first explain the generic message protocol.

The generic message protocol is divided into three areas of functionality as described above. For each area of functionality, the invention provides a server interface having a protocol model, message formats and call-control structures, a set of call control primitives, parameter information and protocol for passing parameters, and a call control library.

The In-band Signalling Server Interface

The In-band Signalling Server (IBSS) protocol model includes Request (REQ), Indication (IND), Response (RSP), and Confirmation (CNF). An IBSS_MSG_REQ primitive is used by the Call Control

entity (CCE) when it requires the use of switching engine functionality. An IBSS_MSG_IND primitive is used by the switching engine to notify the CCE of a REQ. An IBSS_MSG_RSP primitive is used by the CCE to acknowledge receipt of an IND. A IBSS_MSG_CNF primitive is used by the switching engine to notify the CCE that REQuested activity has been completed successfully.

The format of IBSS messages includes a header in the form of IBSS_MSG_HDR_S, followed by a variable number of parameters in the form of IBSS_MSG_PRM_S structures stored consecutively, followed by an "end of parameters" indicator. The "C" structure of the header and parameter format is set out at pages 3-2 through 3-4 of "Access Services™ Switching Engine Interface Specification", copyright 1995 Engineering and Business Systems Inc., Shelton, CT, available as "Part No. D-0084-00-000-000, and also found as Appendix A to U.S. patent application Serial Number 08/662,077. For brevity, this document is referred to hereinafter as "Appendix A".

Nineteen call control primitives are provided for the IBSS. These generally include Outseize, Release, Answer, Wink, Flash, Error, Digits, Inseize, and CPA, each coupled with one or more of Request (REQ), Indication (IND), Response (RSP), and Confirmation (CNF) as appropriate. A complete listing of the IBSS call control primitives is set out in Appendix A at pages 3-4 through 3-10.

Some of the call control primitives utilize parameter information which is set out in Appendix A at pages 3-11 through 3-12.

The IBSS also provides a call control library to facilitate packing and unpacking of IBSS API messages. Seven library calls are provided, the details of which are set out in Appendix A at pages 3-12 through 3-21.

The Connection Server Interface

The Connection Server (CONS) protocol model includes Command (CMD), Acknowledge (ACK), Not Acknowledged (NACK), and Indication (IND). A CONS_MSG_CMD primitive is used by the CCE to request the switching engine to construct or tear down a connection. A CONS_MSG_ACK primitive is returned to the CCE by the switching engine to indicate successful processing of a CMD. A CONS_MSG_NACK primitive is returned to the CCE by the switching engine to indicate unsuccessful processing of a CMD. A CONS_MSG_IND primitive is an autonomous event which indicates whether a logical device is in service or out of service.

The format of CONS messages is CONS_MSG_S in which all header and message specific data is included. The "C" structure of the CONS message format is set out in Appendix A at pages 4-2 through 4-3.

Seventeen call control primitives are provided for the CONS. These generally include 2 Way Connection (2_WAY_CONN), 1 Way Connection (1_WAY_CONN), Disconnection (DISC), and Persistent Connection (NAIL_UP), each of which is coupled to one of (CMD), (ACK), and (NACK). Two autonomous primitives are provided inconjunstion with (IND) to indicate in service (CONS_INS_STATE_IND) and out of service (CONS_OOS_STATE_IND). A complete listing of the CONS call control primitives is set out in Appendix A at pages 4-4 through 4-10.

The CONS also provides a call control library to facilitate packing and unpacking of CONS API messages. The library calls are set out in Appendix A at pages 4-12 and 4-13.

The Media Server Interface

The Media Server (MEDS) protocol model includes Command (CMD), Acknowledge (ACK), Not Acknowledged (NACK), and Indication (IND). A MEDS_MSG_CMD primitive is used by the CCE to request the switching engine to perform a media function. A MEDS_MSG_ACK primitive is returned to the CCE by the switching engine to indicate successful processing of a CMD. A MEDS_MSG_NACK

primitive is returned to the CCE by the switching engine to indicate unsuccessful processing of a CMD. A MEDS_MSG_IND is used to notify the CCE that an autonomous event relating to a previous command has occurred.

The format of MEDS messages is MEDS_MSG_S in which all header and message specific data is included. The "C" structure of the MEDS message format is set out in Appendix A at pages 5-2 through 5-3.

Twenty-six call control primitives are provided for the MEDS. These generally include Connect Tone (CONN_TONE), Disconnect Tone (DISC_TONES), Collect Digits (COLL_DIG), Stop Collecting Digits (STP_DIG_COLL), Outpulse Digits (OUTP_DIG), Connect a Recorded Announcement (CONN_ANN), and Disconnect Announcement (DISC_ANN). The MEDS_CONN_TONE_CMD primitive is used to generate a variety of tones such as dialtone, ringback, line busy, reorder (fast busy), intercept, and bong. The result of the Connect Tone command is either MEDS_CONN_TONE_ACK or MEDS_CONN_TONE_NACK. Some tones are continuous until Discontinued. Other tones complete autonomously and completion is indicated by MEDS_TONE_CMPLT_IND. The MEDS_COLL_DIG_CMD is used to initiate collection of MF or DTMF digit tones. The result of the Collect Digits command is either MEDS_COLL_DIG_ACK or MEDS_COLL_DIG_NACK. In addition, when digit collection is completed, the autonomous primitive MEDS_COLL_DIGITS_IND issues. If there is an error in collecting digits, the autonomous primitive MEDS_COLL_ERR_IND issues. The other primitives in the MEDS API operate in a similar manner. A complete listing of the MEDS call control primitives is set out in Appendix A at pages 5-3 through 5-9.

The MEDS also provides a call control library to facilitate packing and unpacking of MEDS API messages. The library calls are set out in Appendix A at pages 5-11 and 5-12.

Specific Switching Engine Interfaces

As mentioned above, the invention provides a separate switching engine for each brand of digital switch. The switching engines bridge communications between the development system and different switches. Each of the switching engines provides the Servers described above so that the development system communicates with each switching engine in the same manner. Therefore, each switching engine is provided with appropriate translators to convert IBSS, CONS, and MEDS API messages to native switch messages and vice-versa. In addition, each switching engine provides a Managed Object Server which contains all of the logic necessary for the configuration, initialization, and maintenance of a particular type of switch. Utilizing the API described above and the published specifications for an digital switch, one skilled in the art can create a switching engine for use with the development system described herein and in the parent application. The Appendices attached hereto include the specifications for two switching engines according to the invention which are designed to control two popular families of digital switches.

The Summa Four Switching Engine Interface

The details of the Summa Four Switching Engine Interface are set out in Appendix A pages A-1 through A-141. The Summa Four Family of Specialty Digital Switching (SDS) platforms utilize programmable inpulse/outpulse rules and answer supervision templates for inband signalling, connection commands/reports (voice path control and conference control) for establishing and tearing down connections, voice path control commands for tone signalling, and an SNMP interface for OA&M purposes.

The IBSS implementation in the Summa Four Switching Engine utilizes the SDS programmable inpulse/outpulse rules and answer supervision templates to effect the functionality of the IBSS described above. Each of the IBSS functions are directly supported by corresponding native SDS switch messages. IBSS implementation details are set out in Appendix A at pages A-1 through A-20.

The CONS implementation in the Summa Four Switching Engine utilizes the SDS commands/reports and also provides additional logic to maintain connection status per device so that conference points can be allocated when the need arises. In addition, a per-device transaction state is maintained in order to differentiate between reports which could have multiple meanings. Each of the CONS functions are directly supported by one or more native SDS switch messages. CONS implementation details are set out in Appendix A at pages A-20 through A-37.

The MEDS implementation in the Summa Four Switching Engine utilizes the SDS voice path control to effect the functionality of the MEDS described above. All of the MEDS functions except MEDS_TONE_CMPLT_IND are directly supported by one or more native SDS messages. In the case of the MEDS_TONE_CMPLT_IND primitive, the switching engine starts a timer for certain tones when the MEDS_CONN_TONE_ACK is returned. When the time expires, the MEDS_TONE_CMPLT_IND primitive is returned. MEDS implementation details are set out in Appendix A at pages A-37 through A-61.

The Object Server in the Summa Four Switching Engine utilizes tables in the SDS management information database (MIB). Each managed object in the Object Server is equivalent to an individual table in the MIB. When a configuration command is entered into the MML command interpreter (of the development system), the command is parsed and encoded into an Object Server Request PDU which is passed to Object Server for processing. The switching engine maintains a mapping between command parameters and the MIB table attributes. Based on the mapping, the switching engine builds variable binding lists for SNMP get-set PDUs. A dialog is then initiated between the switching engine and the Summa Four SNMP agent. At the completion of the dialog, results are returned to an MML interpreter. Object Server implementation details are set out in Appendix A at pages A-61 through A-126.

The Excel Switching Engine Interface

The details of the Excel LNX-2000 Switching Engine Interface are set out at pages 1-1 through 1-39 of "Excel Switching Engine Interface Specification", copyright 1995 Engineering and Business Systems, Inc., Shelton, CT, and which may be found as Appendix B of U.S. patent application Serial Number 08/662,077. For brevity, this document is referred to hereinafter as "Appendix B". The LNX-2000 platforms utilize a programmable host interface having host messages to effect the functionality of the API described above.

The IBSS implementation in the LNX-2000 Switching Engine utilizes a direct translation of API messages to and from native LNX-2000 host messages. Details of the IBSS implementation are set out in Appendix B pages 1-1 through 1-5.

The CONS interface is implemented as a direct translation between CONS API primitives and LNX-2000 host messages to provide connection related as well as some maintenance services. In most cases the mapping of messages between the CONS API and the LNX-2000 are one to one. In some cases, however, several native host messages are used to effect a single API message. Details of the CONS implementation are set out in Appendix B pages 1-6 through 1-9.

The MEDS interface is implemented as a direct translation between MEDS API primitives and LNX-2000 host messages to provide media related functionality. Details of the MEDS implementation are set out in Appendix B pages 1-10 through 1-14.

The Object Server in the LNX-2000 Switching Engine utilizes LNX host messages to configure and initialize the LNX-2000. When a configuration command is entered into the MML command interpreter (of the development system), the command is parsed and encoded into an Object Server Request PDU which is passed to Object Server for processing. The switching engine maintains a mapping between command parameters and the LNX host messages. Based on the mapping, the switching engine populates the

corresponding LNX message and sends it to the switch for processing. When a reply or an indication is received from the LNX-2000, the switching engine builds a Reply PDU and sends it back to the MML. Object Server implementation details are set out in Appendix b at pages 1-15 through 1-39.

As mentioned above, the API and switching engines of the invention are particularly well suited for use in an applications development environment such as disclosed in the parent application. The present invention incorporates the features of the development system of the parent application with the features of a "generic" API for controlling a digital switch and a number of switching engines for controlling different switches.

Turning now to Figure 5, an applications development system 200 according to the invention generally includes an application layer 202, an object layer 204 and a resource layer 206. The application layer 202 includes a graphical interface with tools for developing telecommunications service applications. The application layer 202 calls upon the server layer 204 to rapidly perform tasks involving signalling and switching. The server layer is provided with its own programming interface (API) so that new services may be added the system in order to support new applications. The server layer 204 calls on the resource layer 206 to control hardware and to interface with the telecommunications system. The resource layer is also provided with its own programming interface (API) so that new hardware devices and new connection services can be added to the system in order to support new applications created in the application layer via new object servers created in the server layer to support the new hardware devices and new connection services.

As seen in Figure 5, a number of switching engines 206a, 206b, for example, are provided in the resource layer 206 in order to control digital switches as described above. The ISS, CONS, and MEDS Servers described above are provided in the server layer 204. Applications created in the application layer can use

the generic API according to the invention to thereby control several different types of digital switches.

There have been described and illustrated herein several embodiments of methods and apparatus for controlling digital communications switching equipment. While particular embodiments of the invention have been described, it is not intended that the invention be limited thereto, as it is intended that the invention be as broad in scope as the art will allow and that the specification be read likewise. Thus, while particular switching engines have been disclosed for use with particular digital switches, it will be appreciated that other switching engines could be created utilized according to the methods of the invention. Also, while particular sets of generic messages have been shown, it will be recognized that other sets of generic messages could be created and utilized with similar results obtained. Moreover, while particular configurations have been disclosed in reference to a development environment incorporating the methods and apparatus of the invention, it will be appreciated that other configurations could be used as well. Furthermore, it will be understood that different development environments can utilize the methods and apparatus disclosed herein. In addition, while the invention has been disclosed with reference to certain host programmable switches, it will be appreciated that the invention can also be used with SCSA, MVIP, and other standards for computer switching on a card rather than in a separate switch which is controlled by a host through a serial interface. Thus, as used herein, the term "switch" includes both a card for a computer and a separate switch coupled to a host. It will therefore be appreciated by those skilled in the art that yet other modifications could be made to the provided invention without deviating from its spirit and scope as so claimed.

Claims:

1. A method for controlling a plurality of different high speed digital telecommunications switches with a single messaging protocol, said method comprising:

a) determining a global set of switch functions to be controlled;

b) categorizing at least some switch functions into first subsets of the global set;

c) defining a set of generic messages for each first subset of switch functions;

d) providing a generic message interpreter for each different switch to interpret generic messages and native switch messages;

e) coupling a first generic message interpreter to a first respective switch; and

f) coupling the first generic message interpreter to a source of generic messages, wherein

messages from the source of generic messages are interpreted by the first generic message interpreter to control the first switch with native switch messages.

2. A method according to claim 1, wherein:

the first subsets of the global set include inband signalling, connection services, and media services.

3. A method according to claim 1, further comprising:

g) categorizing at least some switch functions into a second subset of the global set; and

h) defining a superset of messages for each second subset.

4. A method according to claim 3, wherein:

the first subsets of the global set include inband signalling, connection services, and media services, and

the second subset of the global set includes operation, administration, and maintenance.

5. A method according to claim 4, further comprising:

i) providing each generic message interpreter with an operation, administration, and maintenance interpreter.

6. A method according to claim 1, wherein:

the source of generic messages is an applications development system.

7. A method according to claim 6, wherein:

the applications development system includes

i) at least one man/machine interface (MMI) agent;

ii) an object server with predefined managed objects and a database management library;

iii) means for permitting a developer to create and define a user-defined managed object for the object server;

iv) an object server applications programmer interface (API) means coupled to the at least one MMI agent and coupled to the object server for hiding the internal architecture of the object server from the at least one MMI agent with respect to the predefined and user-defined managed objects; and

v) a database which stores managed object related data, wherein,

the user-defined managed objects utilize the database management library and the database which stores managed object related data, and provides the object server with user-defined managed objects without requiring the object server API to be rewritten.

8. A method according to claim 7, wherein:

at least one generic message interpreter is a predefined managed object.

9. A method according to claim 8, wherein:

said database includes data relating to native switch messages.

10. A method according to claim 1, wherein:

at least some messages from the source of generic messages are interpreted by the first generic message interpreter as multiple native switch messages.

11. An apparatus for controlling a plurality of different high speed digital telecommunications switches with a single messaging protocol, said apparatus comprising:

a) at least one man/machine interface (MMI) agent;

b) an object server with predefined managed objects and a database management library;

c) an object server applications programmer interface (API) means coupled to said at least one MMI agent and coupled to said object server for hiding the internal architecture of the object server from said at least one MMI agent with respect to said predefined managed objects; and

d) a database which stores managed object related data, wherein

said API includes a set of generic messages for controlling the plurality of different high speed digital telecommunications switches,

said object server includes a generic message interpreter, and

said database includes data relating to native switch messages.

12. An apparatus according to claim 11, wherein:

said generic messages are arranged in subsets including messages for inband signalling, connection services, and media services.

13. An apparatus according to claim 12, wherein:

said database includes a superset of native switch messages for operation, administration, and maintenance.

14. An apparatus according to claim 13, wherein:

said API includes a superset of messages for operation, administration, and maintenance.

15. An apparatus according to claim 11, wherein:

said generic message interpreter interprets at least one of said generic messages as multiple native switch messages.

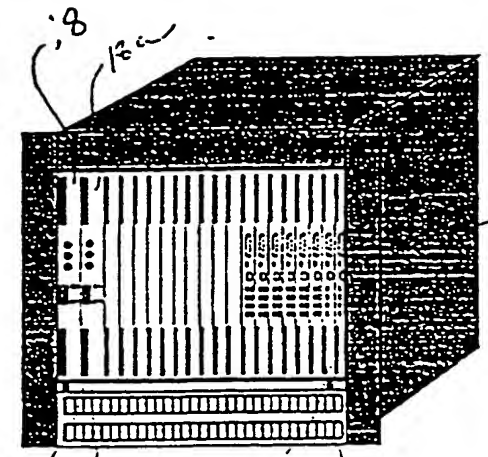



Fig. 1 (Prior Art)

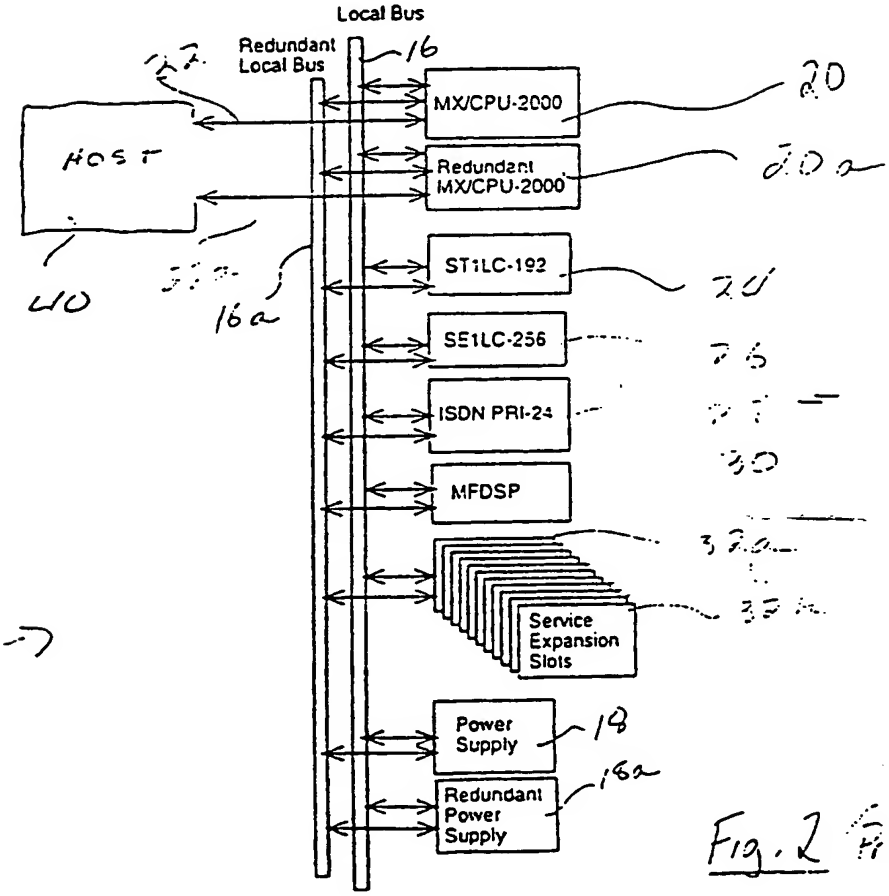



Fig. 2 (Prior Art)

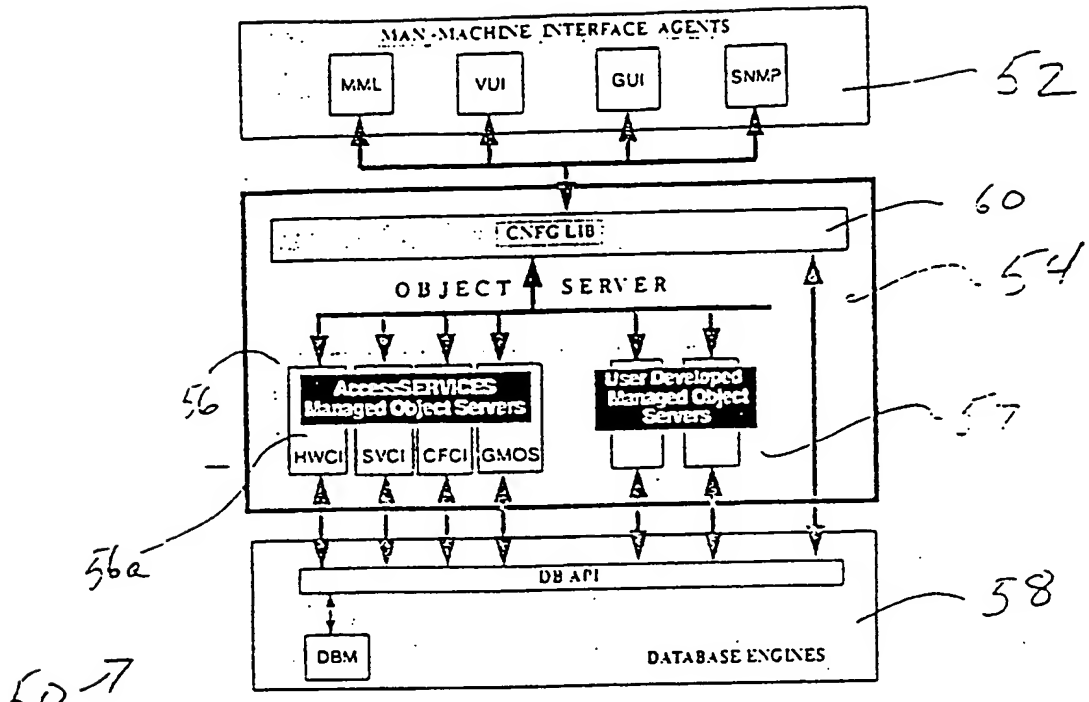

MAN-MACHINE INTERFACE AGENTS
MML
VUI
GUI
SNMP
52
60
CNFG LIB
54
OBJECT SERVER
56
AccessSERVICES Managed Object Servers
HWCI
SVCI
CFCI
GMOS
User Developed Managed Object Servers
57
56a
DB API
58
DBM
DATABASE ENGINES
50

Fig. 3

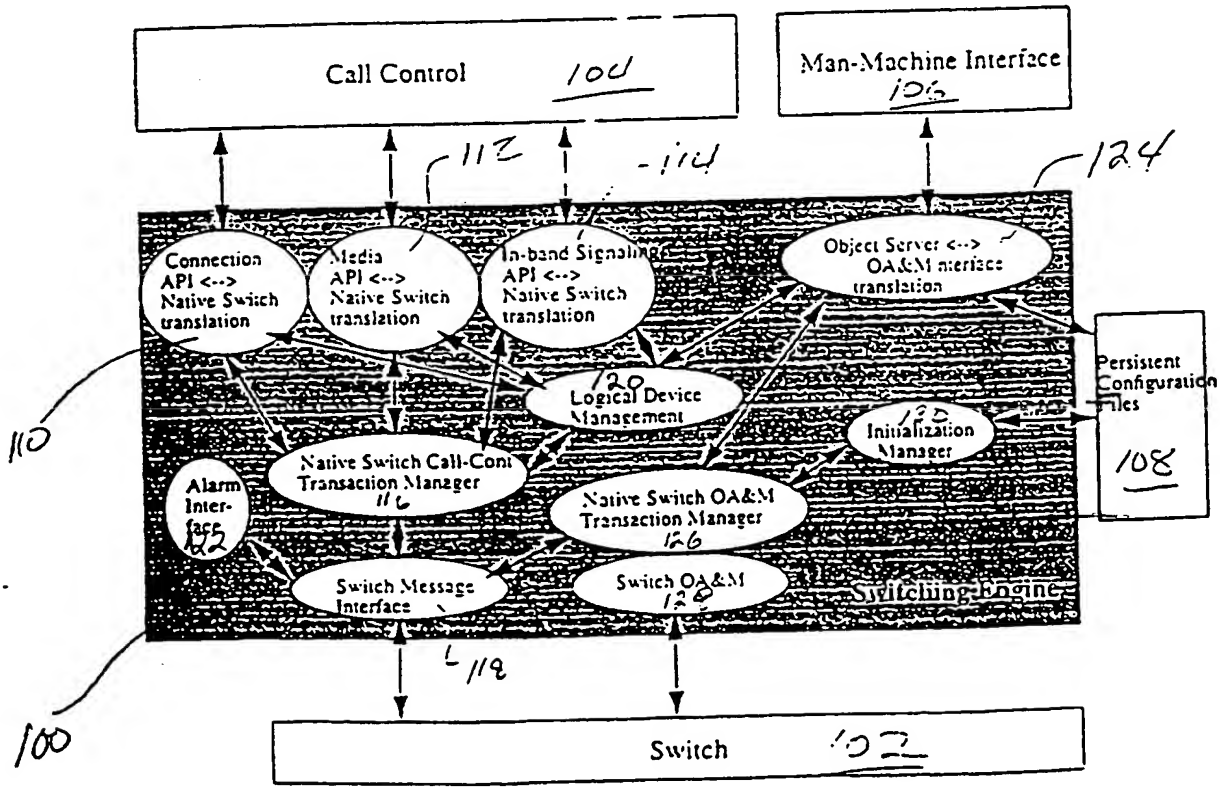

Call Control
100
Man-Machine Interface
106
112
114
124
Connection API <--> Native Switch translation
Media API <--> Native Switch translation
In-band Signaling API <--> Native Switch translation
Object Server <--> OA&M interface translation
120
Logical Device Management
Persistent Configuration Files
108
Initialization Manager
Native Switch Call-Cont Transaction Manager
116
Alarm Inter-face
122
Native Switch OA&M Transaction Manager
126
Switch Message Interface
Switch OA&M
128
Switching Engine
118
110
100
Switch
102

Fig. 4

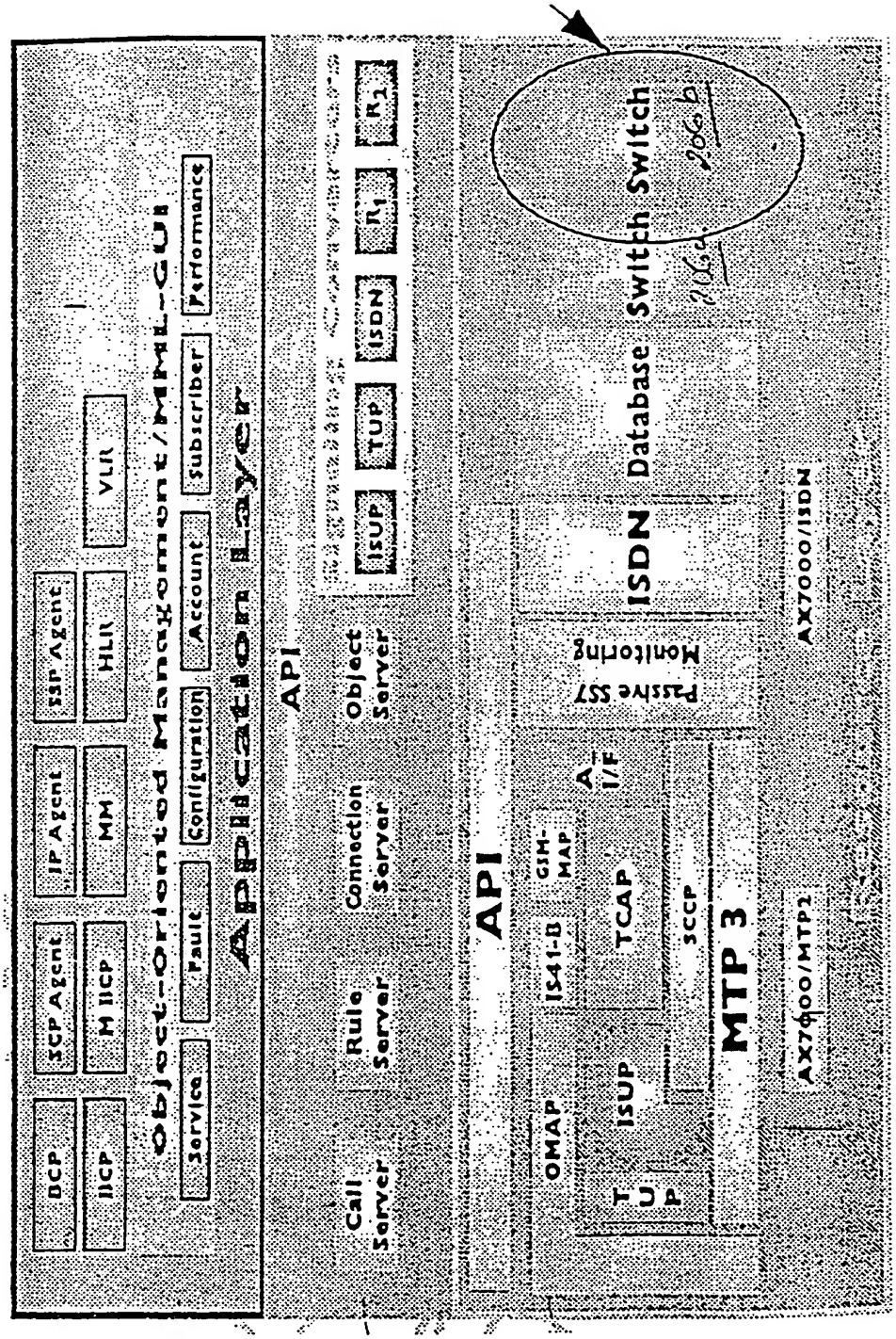

DCP
SCP Agent
IP Agent
SSP Agent
IICP
M IICP
MM
HLR
VLR
Object-Oriented Management/MMI-GUI
Service
Fault
Configuration
Account
Subscriber
Performance
Application Layer
API
Call Server
Rule Server
Connection Server
Object Server
ISUP
TUP
ISDN
R1
R2
API
OMAP
IS41-B
GSM-MAP
TCAP
SCCP
ISUP
TUP
A I/F
MTP 3
Passive SS7 Monitoring
ISDN Database Switch
AX7000/MTP2
AX7000/ISDN
206b
206a
202
204
206
200

Fig. 5

**PCT** WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/00

A3

(11) International Publication Number: WO 97/48234

(43) International Publication Date: 18 December 1997 (18.12.97)

(21) International Application Number: PCT/US97/09996

(22) International Filing Date: 9 June 1997 (09.06.97)

(30) Priority Data:
08/662,077 12 June 1996 (12.06.96) US

(71) Applicant: NEWNET, INC. [US/US]; 2 Enterprise Drive, Shelton, CT 06484 (US).

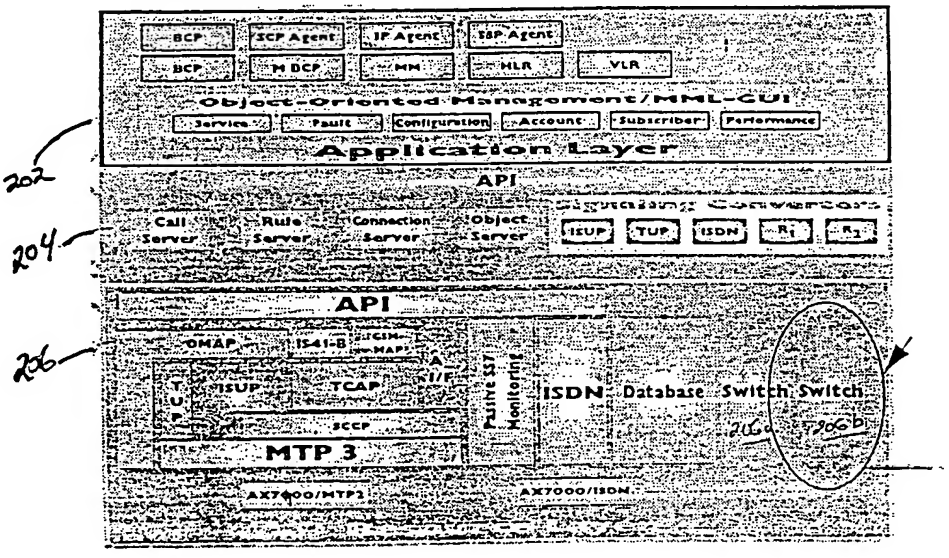
(72) Inventors: HARTMANN, Curtis; 308 Shore Drive, Branford, CT 06405 (US). DUMAN, Osman, Ali; 30 Wake Robin Lane, Shelton, CT 06484 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

**Published**
*With international search report.*

(88) Date of publication of the international search report:
2 July 1998 (02.07.98)

(54) Title: METHODS AND APPARATUS FOR CONTROLLING DIGITAL COMMUNICATIONS SWITCHING EQUIPMENT




(57) Abstract

A generic telecommunications switch messaging protocol and object oriented development system is disclosed for message handling and switch supervision of switching engines conversant with both generic and specific switching protocols. Certain switch messages that are specific to a switch, such as initialization and maintainance messages, are not genericized because their functionality differs from other switches. Figure 5 shows the object oriented applications development system including an application layer (202), an object layer (204), a resource layer (206), a number of switching engines (206A, 206B) are provided in the resource layer (206) to control digital switches. Specific data files are provided for automatic download to the switch and specific MML commands are provided for interpreting configuration commands. Some commands or messages not supported by a particular switch can be supported in the API by providing the intelligence to combine native switch messages to emulate functionality not directly provided by the switch.

*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | |
|---|---|
| AL | Albania |
| AM | Armenia |
| AT | Austria |
| AU | Australia |
| AZ | Azerbaijan |
| BA | Bosnia and Herzegovina |
| BB | Barbados |
| BE | Belgium |
| BF | Burkina Faso |
| BG | Bulgaria |
| BJ | Benin |
| BR | Brazil |
| BY | Belarus |
| CA | Canada |
| CF | Central African Republic |
| CG | Congo |
| CH | Switzerland |
| CI | Côte d'Ivoire |
| CM | Cameroon |
| CN | China |
| CU | Cuba |
| CZ | Czech Republic |
| DE | Germany |
| DK | Denmark |
| EE | Estonia |
| ES | Spain |
| FI | Finland |
| FR | France |
| GA | Gabon |
| GB | United Kingdom |
| GE | Georgia |
| GH | Ghana |
| GN | Guinea |
| GR | Greece |
| HU | Hungary |
| IE | Ireland |
| IL | Israel |
| IS | Iceland |
| IT | Italy |
| JP | Japan |
| KE | Kenya |
| KG | Kyrgyzstan |
| KP | Democratic People's Republic of Korea |
| KR | Republic of Korea |
| KZ | Kazakstan |
| LC | Saint Lucia |
| LI | Liechtenstein |
| LK | Sri Lanka |
| LR | Liberia |
| LS | Lesotho |
| LT | Lithuania |
| LU | Luxembourg |
| LV | Latvia |
| MC | Monaco |
| MD | Republic of Moldova |
| MG | Madagascar |
| MK | The former Yugoslav Republic of Macedonia |
| ML | Mali |
| MN | Mongolia |
| MR | Mauritania |
| MW | Malawi |
| MX | Mexico |
| NE | Niger |
| NL | Netherlands |
| NO | Norway |
| NZ | New Zealand |
| PL | Poland |
| PT | Portugal |
| RO | Romania |
| RU | Russian Federation |
| SD | Sudan |
| SE | Sweden |
| SG | Singapore |
| SI | Slovenia |
| SK | Slovakia |
| SN | Senegal |
| SZ | Swaziland |
| TD | Chad |
| TG | Togo |
| TJ | Tajikistan |
| TM | Turkmenistan |
| TR | Turkey |
| TT | Trinidad and Tobago |
| UA | Ukraine |
| UG | Uganda |
| US | United States of America |
| UZ | Uzbekistan |
| VN | Viet Nam |
| YU | Yugoslavia |
| ZW | Zimbabwe |

# INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/09996

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) : G06F 13/00
US CL : 395/311
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/311, 500, 683

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y,E | US 5,691,973 A (RAMSTROM et al) 25 November 1997, col. 38, lines 20,24,61. | 1-10 |
| Y,P | US 5,546,584 A (LUNDIN et al) 13 August 1996, col. 4, line 47, col. 5, line 36, col. 7, lines 15-22., col. 38, lines 20-61. | 1-15 |
| Y,P | US 5,542,070 A (LEBLANC et al) 30 June 1996, col. 3, lines 28-64, col. 8, line 12, col. 9, line 57, col. 23, line 50, col. 24, line 50. | 11-15 |

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

| | | | |
|---|---|---|---|
| * | Special categories of cited documents: | "T" | later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention |
| "A" | document defining the general state of the art which is not considered to be of particular relevance | "X" | document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone |
| "E" | earlier document published on or after the international filing date | "Y" | document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art |
| "L" | document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) | "&" | document member of the same patent family |
| "O" | document referring to an oral disclosure, use, exhibition or other means | | |
| "P" | document published prior to the international filing date but later than the priority date claimed | | |

Date of the actual completion of the international search
22 DECEMBER 1997

Date of mailing of the international search report
25 MAR 1998

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
R. S. ROSENHOLM
Telephone No. (703) 306-2901

Form PCT/ISA/210 (second sheet)(July 1992)*